3η Σειρά Γραπτών Ασκήσεων,

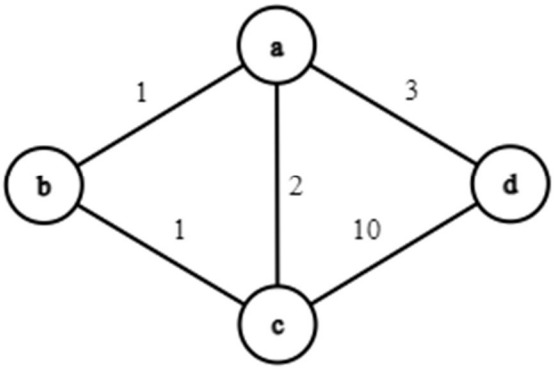
Αλγόριθμοι και Πολυπλοκότητα,

Σεραφείμ Τζελέπης el18849.

Άσκηση 1)

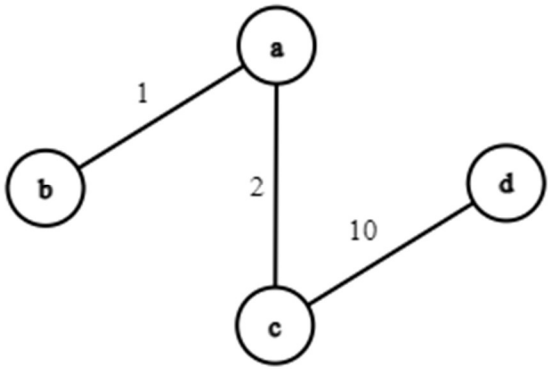
Στην συγκεκριμένη άσκηση ζητείται να υπολογιστεί ένα ελάχιστο συνδετικό δένδρο *T*(s,k)* ενός γραφήματος *G(V,E,w)*, όπου μία συγκεκριμένη κορυφή s πρέπει να έχει βαθμό ίσο με k.

A) Για το πρώτο ερώτημα, αρκεί να βρεθεί ένα αντιπαράδειγμα για το οποίο η άπληστη στρατηγική, σύμφωνα με την οποία στο συνδετικό δέντρο συμπεριλαμβάνονται οι k ακμές μικρότερους βάρους, να μην δίνει την βέλτιστη λύση. Έστω ότι G είναι το εξής γράφημα:

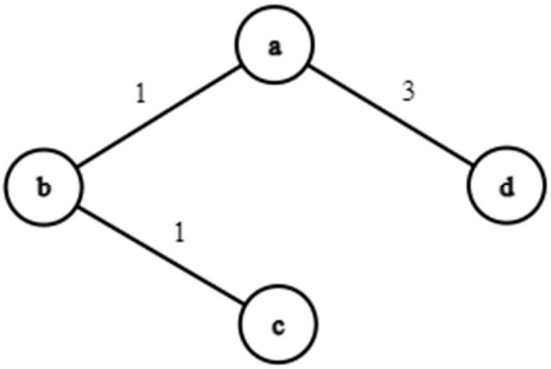


Έστω ότι το ζητούμενο ελάχιστο δένδρο είναι το *T*(a,2),* τότε σύμφωνα με την άπληστη στρατηγική θα συμπεριληφθούν οι ακμές ab και bc η οποίες είναι μικρότερου βάρους, και επειδή ο βαθμός του a δεν μπορεί να υπερβεί το 2 η επόμενη ακμή που θα συμπεριληφθεί θα είναι η cd,

σχηματίζοντας έτσι δέντρο συνολικού βάρους 13, το οποίο φαίνεται παρακάτω.



Όμως το δέντρο το οποίο αποτελείται από τις ακμές ad, ab και bc πληροί επίσης τις προϋποθέσεις του ζητούμενου δέντρου και έχει συνολικό βάρος 5, όπως φαίνεται παρακάτω.



Συνεπώς η άπληστη στρατηγική δεν οδηγεί πάντα στην βέλτιστη λύση.

B) Για τον σχεδιασμό του αλγόριθμου, θα χρησιμοποιηθεί μια επιπλέον σταθερά Μ. Το Μ μπορεί να παίρνει και θετικές και αρνητικές ακέραιες τιμές και η τιμή του προστίθεται σε όλες τις ακμές που προσπίπτουν στον ζητούμενο σχετικό κόμβο. Συνεπώς καθώς μεγαλώνει η τιμή του Μ τόσες λιγότερες προσπίπτουσες ακμές συμπεριλαμβάνονται στο ελάχιστο συνδετικό δέντρο. Επομένως πρέπει να βρεθεί το κατάλληλο Μ τέτοιο ώστε στο ελάχιστο συνδετικό δέντρο ο σχετικός κόμβος να έχει τον ζητούμενο βαθμό. Με δυαδική αναζήτηση μπορεί να βρεθεί η κατάλληλη τιμή του Μ όμως δεν χρειάζεται να ελέγξουμε όλες τις πιθανές ακέραιες τιμές του Μ. Η αναζήτηση θα γίνει για Μ=e-a, για κάθε e ανήκει Ε και για κάθε a ανήκει Α. Το Ε είναι το σύνολο των διαφορετικών βαρών των ακμών του γραφήματος και Α το σύνολο των διαφορετικών βαρών ακμών που προσπίπτουν στον ζητούμενο κόμβο. Αυτό αρκεί καθώς τα μοναδικά Μ για τα οποία αλλάζει ο βαθμός του σχετικού κόμβου στο ελάχιστο συνδετικό δέντρο, σημαίνει πως για κάποιο Μ γίνεται ίση μια προσπίπτουσα στον σχετικό κόμβο με κάποια άλλη του γραφήματος. Αυτό μπορεί να συμβεί μόνο όταν το Μ είναι ίσο με τη διαφορά μιας προσπίπτουσας ακμής και μιας άλλης ακμής του γραφήματος. Για αρκετά αρνητικό Μ το ελάχιστο συνδετικό δέντρο περιλαμβάνει όλες τις προσπίπτουσες ακμές ενώ για αρκετά θετικό περιλαμβάνει τις ελάχιστες δυνατές, συνεπώς εξετάζονται όλες οι τιμές ανάμεσα στην ελάχιστη και την μέγιστη δυνατή. Λαμβάνοντας όλα τα παραπάνω υπόψιν ο αλγόριθμος έχει ως εξής:

1. Ταξινόμηση του συνόλου των Μ, για όλα τα Μ=e-a, για κάθε e ανήκει Ε και για κάθε a ανήκει Α. Η χρονική πολυπλοκότητα της ταξινόμησης αυτής είναι το πολύ $O(|E||A| \log(|E||A|))$.
2. Δυαδική αναζήτηση στο ταξινομημένο σύνολο των Μ. Σε κάθε επανάληψη προστίθεται το εκάστοτε Μ στις προσπίπτουσες ακμές του σχετικού κόμβου και εκτελείται ο αλγόριθμος Kruskal για τον υπολογισμό του ελάχιστου συνδετικού δένδρου. Αν στο δέντρο αυτό ο βαθμός είναι μικρότερος από τον απαιτούμενο το Μ μειώνεται αλλιώς αν είναι μεγαλύτερος από τον απαιτούμενο το Μ αυξάνεται. Η πολυπλοκότητα αυτού του βήματος είναι $O(\log(|E|*|A|))$.

Η συνολική χρονική πολυπλοκότητα είναι *O(|E||A| log(|E||A|))* + *O(log(|E|*|A|))*.

Άσκηση 2)

A) Ο αλγόριθμος που θα υλοποιηθεί ακολουθεί παρόμοια λογική με αυτήν του αλγόριθμου του Bellman-Ford. Αρχικά θα φτιάξουμε ένα πίνακα A[n] όπου n το πλήθος των κορυφών του γραφήματος G. Αρχικοποιούμε τον πίνακα με A[u] = - ∞ για κάθε u ανήκει V εκτός από το u=s όπου η αρχικοποίηση γίνεται ως A[s] = r. Οπότε παρομοίως, με τον Bellman-Ford, ξεκινάμε από την πηγή και για κάθε ακμή a->b ελέγχουμε αν A[a] + p[b] > 0 το δωμάτιο είναι προσβάσιμο και αν A[a] + p[b] > A[b] τότε ανανεώνουμε το A[b] = A[a] + p[b]. Δηλαδή ψάχνουμε την καλύτερη δυνατή(μεγαλύτερη θετική δηλαδή) κατάσταση στην οποία μπορεί να βρεθεί ο παίκτης στο εκάστοτε δωμάτιο b. Αυτή η επαναλαμβανόμενη χαλάρωση στις ακμές συμβαίνει με παρόμοια διαδικασία του Bellman-Ford και συνεπώς η χρονική πολυπλοκότητα του αλγόριθμου είναι *O(|V||E|)*. Στο τέλος της επαναληπτικής διαδικασίας αρκεί να ελέγξουμε αν A[t] > 0.

B) Για το συγκεκριμένο ερώτημα εκτελούμε κανονικά την διαδικασία όπως εξηγήθηκε στο ερώτημα Α. Μετά την ολοκλήρωση της διαδικασίας ελέγχουμε αν A[t]>0.Αν ναι τότε η διαδρομή είναι ασφαλής. Αν όχι, τότε χαλαρώνουμε τις ακμές άλλη μια φορά και αν κάποια τιμή αλλάξει, τότε υπάρχει θετικός κύκλος προσβάσιμος από την πηγή s, με αρχική ζωή r. Έτσι μπορούμε να εντοπίσουμε του κύκλους και να ελέγξουμε αν η κορυφή είναι προσβάσιμη από κάποιο κορυφή αυτού του κύκλου, όπου και σε αυτήν την περίπτωση ο γράφος είναι ασφαλής. Η χρονική πολυπλοκότητα του βήματος αυτού είναι της ίδιας τάξης δηλαδή *O(|V||E|).*

C) Αθροίζουμε όλα τα αρνητικά βάρη του γράφου και έστω ότι αυτό το άθροισμα έχει την τιμή Μ. Γνωρίζουμε ότι η χειρότερη περίπτωση είναι το μοναδικό μονοπάτι προς την πηγή να έχει όλα τα αρνητικά βάρη συνεπώς θα θέλουμε r=M. Συνεπώς αρκεί στην χειρότερη περίπτωση το ελάχιστο r

θα ισούται με M. Στην καλύτερη περίπτωση θα ισούται με 0 καθώς το μονοπάτι θα είναι ούτως ή άλλως θετικό. Συνεπώς για την εύρεση της ελάχιστης τιμής του r για την οποία ο Λαβύρινθος G είναι r-ασφαλής αρκεί να κάνουμε δυαδική αναζήτηση του r στο διάστημα [0,M] και κάθε φορά να εκτελούμε τον αλγόριθμο που υλοποιήσαμε όπου αποφαίνεται αν ο λαβύρινθος είναι r-ασφαλής για δεδομένο r. Κάθε φόρα που ο λαβύρινθος είναι r-ασφαλής η δυαδική αναζήτηση θα περιορίζεται σε μικρότερα r ενώ κάθε φόρα που αποτυγχάνει η δυαδική αναζήτηση θα περιορίζεται στα μεγαλύτερα r. Έτσι με την εξαντλητική χρήση της δυαδικής αναζήτησης και του υλοποιημένου αλγόριθμου μπορεί να υπολογιστεί η ελάχιστη τιμή του r για την οποία ο λαβύρινθος G είναι r-ασφαλής με χρονική πολυπλοκότητα $O((|V|+|E|) \log M)$.

Άσκηση 3)

1) Για την επίλυση του συγκεκριμένου ερωτήματος θα χρησιμοποιούμε greedy approach. Αρχικά, πρέπει να ταξινομήσουμε τις πόλεις βάση την απόσταση τους, έτσι ώστε να το μονοπάτι να είναι διατρέξιμο σε γραμμικό χρόνο. Έπειτα θα χρησιμοποιήσουμε την δομή της στοίβας με σκοπό τον υπολογισμό ενός πίνακα έστω nextStation με μήκος όσο και το πλήθος των σταθμών και αρχικοποιήμενο στο -1 που έχουμε στον οποίο θα έχουμε την θέση του φθηνότερου σταθμού ο οποίος βρίσκεται μετά από αυτόν που βρισκόμαστε. Αυτό θα γίνει διατρέχοντας το μονοπάτι ανάποδα, αρχικώς με άδεια στοίβα, σε κάθε πόλη βρισκόμαστε κοιτάμε το τελευταίο στοιχείο της στοίβας. Αν είναι άδεια η στοίβα τότε δεν υπάρχει σταθμός μεταγενέστερος με φθηνότερο κόστος βενζίνης. Αν έχει άλλο στοιχείο/πόλη ελέγχουμε αν το κόστος της βενζίνης είναι φθηνότερο σε εκείνη την πόλη. Αν ναι τότε αποτελεί την φθηνότερη μεταγενέστερη πόλη αλλιώς την βγάζουμε από την κορυφή της στοίβας και ελέγχουμε το επόμενο στοιχείο της. Στην συνέχεια προσθέτουμε το σταθμό που βρισκόμαστε στην κορυφή της στοίβας. Έτσι έχουμε συνολικά σε O(n) τον πίνακα nextStation που εξηγήθηκε προηγουμένως. Συνεπώς πλέον έχοντας τον nextStation έτοιμο μπορούμε να ξεκινήσουμε την τελική επίλυση. Διατρέχουμε το s-t μονοπάτι από την αρχή προς το τέλος. Σε κάθε σταθμό που

βρισκόμαστε αποφασίζουμε πόση βενζίνη θα βάλουμε. Αυτή η απόφαση γίνεται ως εξής. Αν υπάρχει nextStation[current] $\neq -1$ δηλαδή υπάρχει σταθμός πριν το τέλος με κόστος βενζίνης μικρότερο από αυτού που βρισκόμαστε τότε ελέγχουμε αν είναι προσβάσιμος με απόσταση μικρότερη του B. Αν ναι τότε βάζουμε όσα λίτρα χρειάζονται για να φτάσουμε σε εκείνον τον σταθμό. Αλλιώς γεμίζουμε το ντεπόζιτο. Αν το nextStation[current] = -1 τότε δεν υπάρχει κόμβος με μικρότερο κόστος βενζίνης μέχρι το τέλος. Συνεπώς αν ο τελικός κόμβος είναι μικρότερης απόστασης από B βάζουμε βενζίνη μέχρι να φτάσουμε συμπληρώσουμε τα λίτρα για να φτάσουμε τον τελικού κόμβο. Αν ο τελικός κόμβος απέχει παραπάνω από B φουλάρουμε το ντεπόζιτο. Για την εύρεση των αποστάσεων σε O(1) αρκεί να υλοποιήσουμε έναν prefix array σε O(n). Το να διατρέξουμε το μονοπάτι γίνεται σε O(n). Η αρχική ταξινόμηση είναι πολυπλοκότητας O(nlogn). Άρα η συνολική χρονική πολυπλοκότητα είναι O(nlogn).

2) Για την εύρεση του ελάχιστου κόστους για να πάμε από το s στο t σε ένα γράφημα G και όχι σε ένα συγκεκριμένο μονοπάτι θα χρειαστεί να προσεγγίσουμε το πρόβλημα με την λογική του δυναμικού προγραμματισμού. Έστω D(u,g) το ελάχιστο κόστος να πάμε από τον κόμβο/πόλη u στο t ξεκινώντας με βενζίνη g. Συνεπώς το ζητούμενο κόστος σε εμάς είναι το D(s,0). Ο αναδρομικός τύπος για τον υπολογισμό του D(u,g) είναι :

$$\boldsymbol{D(u,g)}$$
$$\boldsymbol{= min_{duv \le U}} \begin{cases} \boldsymbol{D(v,0) + (d_{uv} - g)c(u)} : \boldsymbol{c(v) \le c(u)\ and\ g \le\ d_{uv}} \\ \boldsymbol{D(v, U - d_{uv}) + (U - g)c(u)} : \boldsymbol{c(v) \ge c(u)} \end{cases}$$

Θέλουμε να αποτυπώσουμε γραφικά όλους τους υπολογισμούς και τις περιπτώσεις που ελέγχουμε βάσει του αναδρομικού υπολογισμού του D(s,0). Συνεπώς δημιουργείται ένα νέο γράφημα έστω H το οποίο έχει ως κορυφές το εκάστοτε ζεύγος u,g και ακμές με βάρος το οποίο προκύπτει από τις τιμές από τον υπολογισμό του D(v,g) για τις αποδεκτές πόλεις v. Έτσι, το γράφημα αυτό έχει όλες τις πιθανές κινήσεις, και συνεπώς αν βρούμε το ελάχιστο μονοπάτι ουσιαστική

αποτελεί την βέλτιστη διαδρομή s-t. Η χρονική πολυπλοκότητα του υπολογισμού μέσω της εξίσωσης δυναμικού προγραμματισμού πολυπλοκότητας $O(n^2)$. Για τον Dijkstra, έχουμε n^2 κορυφές και n^3 ακμές. Άρα έχουμε συνολική πολυπλοκότητα $O(n^3 + n^2 \log n^2) = O(n^3)$.

Άσκηση 4)

Πριν ξεκινήσουμε να υλοποιούμε τον αλγόριθμο μας, πρέπει να κάνουμε μια παρατήρηση. Παρατηρούμε ότι για να αιχμαλωτιστεί ένας συγκεκριμένος εξτρεμιστής ή να καταστραφεί μια βάση από τον στρατό το ελάχιστος κόστος επιτυγχάνεται από τον στρατιώτη που έχει την πιο κοντινή θέση, εφόσον γνωρίζουμε επιπλέον ότι κατόπιν της αιχμαλώτισης ή καταστροφής θα επιστρέψει στην αρχική του θέση. Μπορεί συνεπώς το πρόβλημα αυτό να αναχθεί σε πρόβλημα εύρεσης της ελάχιστης τομής ισοδύναμο με αυτό της μέγιστης ροής. Αρχικώς κατασκευάζουμε ένα γράφημα G με m + k + 2 κόμβους, οπού οι m αντιστοιχούν στους εξτρεμιστές και οι k αντιστοιχούν στις βάσεις. Έχουμε επιπλέον 2 κόμβους s και t οι οποίοι αντιστοιχούν σε source και sink του γραφήματος. Για κάθε κόμβο που αντιστοιχεί σε Εξτρεμιστή τον συνδέουμε με κάθε κόμβο βάσης από την οποία η απόσταση είναι $\leq$ d. Κάθε τέτοια ακμή έχει άπειρη χωρητικότητα. Στην συνέχεια συνδέουμε κάθε κόμβο εξτρεμιστή με τον κόμβο s και χωρητικότητα το ελάχιστο κόστος του στρατού για να αιχμαλωτίσει τον αντίστοιχο εξτρεμιστή δηλαδή την απόσταση του κοντινότερου στρατιώτη σε αυτόν. Τελικά συνδέουμε κάθε κόμβο ο οποίος αντιστοιχεί σε κόμβο βάσης με τον κόμβο t με χωρητικότητα το ελάχιστο κόστος του στρατού για να καταστρέψει την βάση. Επομένως, παρατηρούμε ότι το min cut s-t στο γράφημα G αποτελεί το ζητούμενο κόστος καθώς θα περιέχει μόνο ακμές από το s σε κόμβους που αντιστοιχούν σε εξτρεμιστές και από ακμές από το t σε κόμβους που αντιστοιχούν σε βάσεις. Συνεπώς βρίσκουμε τον βέλτιστο συνδυασμό επιθέσεων(ελάχιστο κόστος) για να αιχμαλωτιστούν όλοι οι εξτρεμιστές είτε αυτό συμβαίνει με άμεση επίθεση σε αυτούς είτε με επιθέσεις σε κάποιες βάσεις Για την δημιουργία του γραφήματος G πρέπει να ταξινομηθούν οι βάσεις, οι στρατιώτες και οι εξτρεμιστές βάση τη θέση τους στην γραμμή. Η χρονική πολυπλοκότητα της ταξινόμησης αυτής είναι $O((n+k+m)\log(n+k+m))$. Η εύρεση

του s-t min cut μπορεί να βρεθεί από s-t max flow. Για την εύρεση του max flow θα χρησιμοποιήσουμε τον αλγόριθμο Dinic's ο οποίος έχει χρονική πολυπλοκότητα $O((|V|^2|E|)$ όπου V = m + k + 2 και E το πλήθος των ακμών του G.

Άσκηση 5)

Τακτοποίηση Ορθογωνίων Παραλληλογράμμων:

Έστω ορθογώνια A_i με μεγέθη αi × ϵ, με $\epsilon \ll 1$ που μπορεί να επιλεγεί κατάλληλα. Έστω B μεγάλο ορθογώνιο μεγέθους t x 2ϵ. Πρέπει να επιλέξουμε t = $t = \frac{1}{2}\sum_i a_i$ και έτσι να τοποθετήσουμε τα n ορθογώνια μέσα στο B. Τότε λόγω πλάτους θα είχαμε κάνει partition το σύνολο ο {α1, . . . , αn} σε δύο σύνολα ίσου μεγέθους. Επομένως, πρέπει να επιλέξουμε το ϵ κατάλληλα ώστε να μην επιτρέψουμε περιστροφές στην τοποθέτηση των Ai. Έτσι το πρόβλημα αυτό ανάγεται από το 2-partition.

Μέγιστη Τομή με Βάρη στις Κορυφές:

Το πρόβλημα αυτό είναι πολυωνυμικά ισοδύναμο με την μεγιστοποίηση του βάρους της τομής. Επιπλέον κάθε βάρος w(v) ≥ 1, οπότε αν βρεθεί partition (S, V \ S) των κορυφών με τομή τουλάχιστον βάρους B θα έχουμε :

$$B \leq w(S, V\backslash S) = \sum_{u \in S, v \notin S} w_u w_v = \sum_{u \in S} w_u \sum_{v \in n\backslash S} w_v$$

Αν $A = \sum_{i \in [n]} w_i$ τότε το δεξί μέλος της ανισότητας μπορεί να γραφθεί ως :

$$w(S)\big(A - w(S)\big) \leq \frac{A^2}{4}$$

Αυτό προκύπτει, καθώς αν βρούμε το μέγιστο της κοίλης συνάρτησης $f(x) = x(A - x)$ αυτό βρίσκεται στο $x = A/2$. Συνεπώς μπορούμε να δουλέψουμε με το πρόβλημα 2-Partition. Αν ένα σύνολο είναι θετικό και θέλουμε να το διαμερίσουμε σε δύο ισοβαρή υποσύνολα ,τότε, από την ανάλυση που προηγήθηκε, αρκεί να μεγιστοποιήσουμε την τομή ενός γραφήματος με κορυφές όσες και ο πληθάριθμος του 2-Partition instance και βάρη στις κορυφές ίσα με τα στοιχεία του δοθέντος συνόλου.

Αραιό Γράφημα:

Γνωρίζουμε ότι το Independent Set είναι NP-complete πρόβλημα. Έστω γράφημα G = (V,E) με $g \in Z$ και προκύπτει $S \subset V$ με |S|$\geq g$ και $\forall x, y \in S, (x, y) \notin E$ δηλαδή δεν υπάρχουν ακμές μεταξύ των κορυφών του S. Για να αναχθεί το πρόβλημα του independent set σε Sparse Subgraph πρώτα πρέπει να δείξουμε ότι το independent set να μετατραπεί σε input για SS. Έστω k = g και l = 0. Για έναν δοσμένο Independent Set S τότε αποτελεί και ένα αραιό γράφημα(SS) καθώς δεν υπάρχουν ακμές μεταξύ κορυφών του S και |S| = g = k.

Συντομότερο Μονοπάτι με περιορισμούς:

Θεωρούμε ότι έχουμε ένα instance για το decision knapsack πρόβλημα με n αντικείμενα με κόστος c_i και βάρος w_i. Για κάθε αντικείμενου αυτού του instance δημιουργούμε ένα vertex σε έναν πλήρη κατευθυνόμενο γράφο. Κάθε ακμή που φεύγει από το vertex θα έχει βάρος w_i και κόστος c_i. Χρησιμοποιούμε αυτό ως input στο πρόβλημα μας. Έστω ότι έχουμε λύση σε πολυωνυμικό χρόνο. Αν αυτή υπάρχει τότε έχουμε λύση για το knapsack πρόβλημα σε πολυωνυμικό χρόνο γιατί το μονοπάτι s-t περνάει μία φόρα από κάθε vertex και ακμή. Έτσι αποδείξαμε ότι μπορεί να λυθεί σε πολυωνυμικό χρόνο, κάτι που δεν ισχύει.